View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

ELSEVIER

The Journal of Logic Programming 40 (1999) 185–213

THE JOURNAL OF LOGIC PROGRAMMING

Numerical reasoning with an ILP system capable of lazy evaluation and customised search

Ashwin Srinivasan[a,*], Rui Camacho[b]

[a] *Oxford University Computing Laboratory, Wolfson Building, Parks Road,Oxford OX1 3QD, UK*
[b] *LIACC-CIUP, R. Campo Alegre, 4150 Porto, Portugal*

Received 16 September 1996; received in revised form 4 March 1997; accepted 12 January 1999

Abstract

Using problem-specific background knowledge, computer programs developed within the framework of Inductive Logic Programming (ILP) have been used to construct restricted first-order logic solutions to scientific problems. However, their approach to the analysis of data with substantial numerical content has been largely limited to constructing clauses that: (a) provide qualitative descriptions (''high'', ''low'' etc.) of the values of response variables; and (b) contain simple inequalities restricting the ranges of predictor variables. This has precluded the application of such techniques to scientific and engineering problems requiring a more sophisticated approach. A number of specialised methods have been suggested to remedy this. In contrast, we have chosen to take advantage of the fact that the existing theoretical framework for ILP places very few restrictions of the nature of the background knowledge. We describe two issues of implementation that make it possible to use background predicates that implement well-established statistical and numerical analysis procedures. Any improvements in analytical sophistication that result are evaluated empirically using artificial and real-life data. Experiments utilising artificial data are concerned with extracting constraints for response variables in the text-book problem of balancing a pole on a cart. They illustrate the use of clausal definitions of arithmetic and trigonometric functions, inequalities, multiple linear regression, and numerical derivatives. A non-trivial problem concerning the prediction of mutagenic activity of nitroaromatic molecules is also examined. In this case, expert chemists have been unable to devise a model for explaining the data. The result demonstrates the combined use by an ILP program of logical and numerical capabilities to achieve an analysis that includes linear modelling, clustering and classification. In all experiments, the predictions obtained compare favourably against benchmarks set by more traditional methods of quantitative methods, namely, regression and neural-network. © 1999 Elsevier Science Inc. All rights reserved.

Keywords: Inductive Logic Programming; Numerical reasoning; Lazy evaluation; Customised search

* Corresponding author. Fax: +44-1865-273839; e-mail: ashwin.srinivasan@comlab.ox.ac.uk

0743-1066/99/$ – see front matter © 1999 Elsevier Science Inc. All rights reserved.
PII: S0743-1066(99)00018-7

1. Introduction

The framework defining Inductive Logic Programming (ILP: see Ref. [22]), has seen the advent of efficient, general-purpose programs capable of using domain-specific background knowledge to construct automatically clausal definitions that in some sense, generalise a set of instances. This has allowed a novel form of data analysis in molecular biology [15,16,26], stress analysis in engineering [7], electronic circuit diagnosis [11], environmental monitoring [10], software engineering [1], and natural language processing [49]. Of these, some, such as those described in Refs. [1,10,11,26,49], are naturally classificatory. Others, such as those described in Refs. [7,15,26], are essentially concerned with predicting values of a numerical "response" variable (for example, chemical activity of a compound). For problems of this latter type, ILP programs have largely been restricted to constructing definitions that are only capable of qualitative predictions (for example, "high", "low" etc.). Further, if the definition involves the use of any numerical "predictor" variables, then this usually manifests itself as inequalities that restrict the ranges of such variables. This apparent limitation of ILP programs has been of some concern, and rates highly on the priorities of at least one prominent research programme designed to address the shortcomings of ILP [5].

In theory, any form of numerical reasoning could be achieved from first principles by an ILP program. Thus, much of the limitations stated above must stem from practical constraints placed on ILP programs. Some of these constraints pertain to ILP programs like those described in Refs. [29,33], where background knowledge is restricted to ground unit clauses. But what about programs capable of understanding background knowledge that includes more complex logical descriptions? Such programs are in some sense closer to the spirit of the ILP framework defined in Ref. [22]. In this paper, we explore the possibility of improving the numerical capabilities of such an ILP program by the straightforward approach of including as background knowledge predicates that perform numerical and statistical calculations. In particular, by the phrase "numerical capabilities" we are referring to the ability to construct descriptions that may require at least the following:

- Arithmetic and trigonometric functions.
- Equalities and inequalities.
- Regression models (including equations constructed by linear or non-linear regression).
- Geometric models (that is, planar shapes detected in the data).

An ILP program capable of using such primitives would certainly be able to provide more quantitative solutions to the molecular biology and stress analysis problems cited earlier. In this paper we describe two implementation details that considerably improve the quantitative capabilities of the ILP program. The first allows the inclusion of arbitrary statistical and numerical procedures. The second allows, amongst others, a cost function to be minimised when obtaining predictions. It is important to note that these are implementation details only, and do not in anyway, compromise the general applicability of the ILP program. The capabilities for quantitative analysis are assessed empirically with experiments using artificial and natural data.

Experiments with artificial data are concerned with extracting constraints – in the form of equations – for numerical variables from simulator data records of a control

task. Balancing a pole on a cart is a text-book problem in control engineering, and has been a test-bed for evaluating the use of machine learning programs to extract comprehensible descriptions summarising extensive simulator records of controller behaviour. Data records are usually tabulations of the values of numerical-valued variables, and so far, feature-based machine learning programs either equipped with built-in definitions for inequalities or those capable of regression-like behaviour have been used to analyse such data. There are some advantages to the pole-and-cart problem. First, the simulations provide ready access to data records. Second, the nature of the equations to be extracted is relatively straightforward, and known prior to the experiments (from the dynamics of the physical system concerned: see Appendix B). This allows us to focus on the question of whether the ILP program is able to reconstruct these equations.

The experiments with artificial data whilst being instructive, are unrepresentative. In most realistic scenarios, the nature of the underlying model is not known. Under this category, we examine the case of predicting the mutagenic activity of a set of nitroaromatic molecules as reported in Ref. [6]. In that study, the authors identify these compounds as belonging to two disparate groups of 188 and 42 compounds, respectively. The main interest in the group of 42 compounds stems from the fact that they are poorly modelled by the analytic methods used by experts in the field. Elsewhere [16] an ILP program has been shown to find qualitative descriptions for activity amongst some of these molecules, but no models capable of quantitative prediction was reported. The second set of experiments reported in this paper is concerned with constructing an explanation for this data.

The paper is organised as follows. Section 2 introduces the main features of a general ILP algorithm, and how these are implemented within the P-Progol program. It also describes aspects within the P-Progol implementation that impede its use when analysing numerical data. Section 3 describes two general-purpose extensions to the implementation of an ILP algorithm that overcome such problems. Section 4 describes how this work contributes to existing research in this area. Section 5 contains the pole-and-cart experiment, and Section 6 the experiment with predicting mutagenic activity. Section 7 concludes this paper.

2. ILP and P-Progol

2.1. Specification

Following Ref. [23], we can treat P-Progol as an algorithm that conforms to the following partial specifications (we refer the reader to Ref. [19] for definitions in logic programming).

- B is background knowledge consisting of a set of definite clauses $C_1 \wedge C_2 \wedge \ldots$,
- E is a set of examples $= E^+ \wedge E^-$ where:
  - $E^+ = e_1 \wedge e_2 \wedge \ldots$ are "positive examples" that are definite clauses. These are often ground unit clauses,
  - $E^- = \overline{f_1} \wedge \overline{f_2} \wedge \ldots$ are "negative examples" that are Horn clauses. These are also often ground unit clauses, and
  - $B \not\models E^+$ – that is, there is some prior necessity to construct a hypothesis.

- $H = D_1 \wedge D_2 \wedge \ldots$, the output of the algorithm given B and E, is a good, consistent explanation of the examples and is from a predefined language $\mathscr{L}$. Such an output usually satisfies at least the following conditions:
  - Each D_i in H has the property that it can explain at least one positive example. That is, $B \wedge D_i \models e_1 \vee e_2 \vee \ldots$, where $\{e_1, e_2, \ldots\} \subseteq E^+$;
  - $B \wedge H \models E^+$;
  - $B \wedge H \not\models \square$;
  - $B \wedge H \wedge E^- \not\models \square$; and
  - $| B \wedge H | < | B \wedge E^+ |$ where $| \ldots |$ denotes some measure of size.

In practice, P-Progol typically does not meet the requirement for "Strong Consistency", as some members of E^- are treated as "noise". This introduces some complications in the calculation of $| B \wedge H |$, which have to be augmented by an amount required to specify the noisy instances (or exceptions). The hypothesis language $\mathscr{L}$ is specified by:

- Predicate argument annotations (or "modes"). These are usually of the form:
  - *Input/Output/Constant*. An input argument is a variable that is expected to be instantiated, and will not be further instantiated by the predicate. An output argument is a variable that is not expected to be instantiated. This variable will be instantiated by the predicate. For predicates that are "non-deterministic", multiple instantiations may be be possible on backtracking. An argument can be specified as being instantiated to a constant symbol.
  - *Type*. The type or domain of each argument. The set of acceptable values of each type may be specified by enumeration or by an intensional definition.
- Other specifications concerning clauses acceptable for inclusion in the hypothesis. These are usually:
  - *Variable depth*. The maximum length "chain" of input and output variables in a clause.
  - *Inconsistency*. The extent to which the requirement of "Strong Consistency" can be violated by the clauses. This usually is an upper bound on the number of examples in E^- that can be treated as "noise".
  - *Clause length*. The maximum number of literals in any clause.

2.2. Implementation

P-Progol [24] implements the specifications listed above by constructing the set H one clause at a time. Each clause to be added to the set is obtained using an admissible search procedure that evaluates potential candidates along the criteria of sufficiency, consistency, and compression. Complete descriptions of the P-Progol algorithm are available in Ref. [24], and only the main steps are shown in Fig. 1.

The construction of $\perp$ in Step 4 is complicated and we refer the reader to Ref. [24]. For the purposes of this paper, it is sufficient to note that $\perp$ is usually a definite clause (typically with many 100s of literals), and anchors one end of the space of clauses to be searched (the other end being $\square$). Clauses in S_c (Step 5) are enumerated one at a time – starting from $\square$ – by using a built-in refinement operator (in the sense described by Ref. [42], and denoted by ρ). The auxiliary function *bestclause/4* (again, built-in to the implementation) returns the clause adjudged to be the best using some measure of "compression". Two exceptional situations arise. First, there is no clause that passes the test of compression. In this case the example e selected is returned.

1. B, $\mathcal{L}$ are given. $E = E^+ \cup E^-$ are given and have the same predicate symbol as the target relation. Let c be the specification in $\mathcal{L}$ of the maximum number of negative literals allowed.
2. If $E = \emptyset$ then return B.
3. Let e be the first example in E.
4. Construct $\perp$ s.t. $\perp \models \overline{B \cup \{\overline{e}\}}$
5. Let
   $S_c = \{C : \square \succeq [C] \succeq [\perp]$ and C has at most c negative literals $\}$
6. Let $D = \text{bestclause}(B, e, E, S_c)$
7. Let $B = B \cup D$
8. Let $E' = \{e : e \in E^+$ and $B \cup \{\overline{e}\} \vdash \square\}$
9. Let $E = E - E'$.
10. Goto 2.

Fig. 1. The P-Progol algorithm. Here $\succeq$ denotes a subsumption ordering, and [.] denotes an equivalence class.

Second, several clauses have equally good compression. In this case, the first one obtained is returned.

2.3. Shortcomings for numerical reasoning

The implementation, as described in Section 2.2 poses some special difficulties when dealing with the analysis of numerical data. The first, concerns the use of functions whose functional result(s) depend on more than 1 example. For example, univariate regression can be seen as a function, that, given examples of (X,Y) pairs, returns the tuple (M,C) representing the slope and intercept of the line of best-fit through the (X,Y) pairs. A correct computation of (M,C) requires all the (X,Y) pairs. For such cases, the implementation of P-Progol is unable to return clauses containing the correctly computed values. This stems from the fact that clauses are constructed by selecting from a $\perp$ clause constructed from a randomly chosen, *single* example. Some attempt to overcome this can be made by "guessing" correct values of such functional outputs from the example chosen (for example, see Ref. [28]). However, there are obvious limitations to this approach.

The second difficulty in the current implementation arises from a mismatch in the criterion being optimised during clause selection. This criterion, encoded within the *bestclause/4* function in Section 4, is typically a description-length criterion. In dealing with numeric data, the criterion to be minimised is usually different (for example, expected mean-square error). A minor concern pertains to the fact that there may be no concept of "negative" examples when dealing with numerical data. Learning from positive examples only has recently been addressed within the ILP framework in various ways (for example see Refs. [25,38,37]). However, this has not proved a

difficulty for the problems addressed here, and the changes to the implementation described in the next section adequately address these issues. Examples of their form and use are available in Appendix A.

3. Two changes to the implementation

3.1. Lazy evaluation of functions

Using a most-specific clause from a single example to guide the search for clauses prevents P-Progol from using predicates like linear regression where the functional result (here the coefficients of regression) is determined from a set of values that arise from more than one example. To address this, we propose the technique of "lazily evaluating" functional outputs. For a particular function, this requires that the background knowledge B has the following: (1) a mode annotation specifying input and output argument(s) along with their types (in the sense described in Section 2.1); and (2) a definition specifying the procedure for evaluating the output argument(s) from sets of values for the input argument(s). Provided with this, the following modifications to the basic algorithm are implemented:

1. Notionally construct a sentence $\bot_X$ from the most-specific clause $\bot$ constructed in Step 4. $\bot_X$ differs from $\bot$ only in the (negative) literals in $\bot$ that represent lazily-evaluated functions. For these, the corresponding literal in $\bot_X$ is obtained by replacing the output terms with existentially-quantified variables. Each such variable is distinct from any other output variable in $\bot_X$. For example, if $q/2, r/2$ represent lazily-evaluated functions whose second argument is denoted "output", and $\bot$: $\forall A \forall B p(A,B) \leftarrow q(A,3), r(B,3)$ then $\bot_X$: $\forall A \forall B \exists Y \exists Z p(A,B) \leftarrow q(A,Y), r(B,Z)$. $\bot_X$ will take the place of $\bot$ in the search.
2. When incrementally constructing clauses, if the literal selected requires lazy evaluation then its output values are computed (see below). These values will appear in place of the existentially quantified variables introduced into the literal in $\bot_X$. If no output values are obtained, then the literal is not added to the clause.

Without loss of generality, assume that the predicate $q/2$ has been marked for lazy evaluation, and the mode annotations state the first argument to be input and the second as output. This literal now appears as $q(X, Sk_y)$ in $\bot_X$. Here Sk_y is an existentially quantified variable that does not appear as the output of any other literal in $\bot_X$. In the search, we are concerned with adding this literal to a definite clause C. The main steps of computing the output value of $q/2$ are shown in Fig. 2.

For a given h, the procedure in Fig. 2 clearly terminates. Note that the procedure assumes that the background knowledge B is complete to the extent that an answer substitution for Y is obtainable within h resolution steps by the query $\mathcal{Q}([\theta_p, \theta_n], Y)$?. As stated, it is evident that even when several answer substitutions exist for Y, the procedure returns only one. This is not of concern if the literal being evaluated represents a function. We also note in passing that the θ_p^X, θ_n^X are similar to the "positive" and "negative" substitutions defined in Refs. [36,37].

The inclusion of lazy evaluation results in one additional violation to the algorithm described in Fig. 1. There, any clause C in the search is such that $\square \succeq [C] \succeq [\bot]$, where $\succeq$ is a subsumption ordering – normally Plotkin's θ-subsumption [32]. Analogously, with lazy evaluation it would be desirable to show a clause C

1. Let $l = q(X, Sk_y), C = head \leftarrow body$
2. Let $\theta_p^X = \{X/P_1, X/P_2, \ldots\}$ be the substitution instances for X from each substitution θ_{p_i} such that $head \cdot \theta_{p_i} \in E^+$ and $B \vdash_h body \cdot \theta_{p_i}$
3. Let $\theta_n^X = \{X/N_1, X/N_2, \ldots\}$ be substitution instances of X from each substitution θ_{n_i} such that $\overline{head \cdot \theta_{n_i}} \in E^-$ and $B \vdash_h body \cdot \theta_{n_i}$
4. If there is a ground term T_i such that $B \vdash_h \mathcal{Q}([\theta_p^X, \theta_n^X], T_i)$ then $\theta = \{Y/T_i\}$ otherwise $\theta = \emptyset$
5. return θ

Fig. 2. The lazy evaluation procedure. Here h is a natural number, and $\vdash_h$ denotes derivation in at most h resolution steps. $\mathcal{Q}$ is used to distinguish the definition that allows computation of output values for $q/2$.

in the search is such that $\square \succeq_X [C] \succeq_X [\perp_X]$ where $\succeq_X$ is a (possibly different) subsumption ordering. We do not explore this further here.

The technique of lazy evaluation can be extended to handle multiple answer substitutions and even to the construction of new predicate definitions "on-the-fly." However in this paper, its scope will be restricted to the evaluation of functions like linear regression.

3.2. User-defined search

The problem of mismatch in the optimising criterion arises from a more general feature common to most (perhaps all) current programs implementing the ILP specification. This is to do with the fact that a search procedure is built-in to the implementation. We propose to remedy this by allowing the search for clauses to be specified as part of the background knowledge B. This requires Prolog definitions for at least the following:

1. *Refine*. Definitions for a refinement operator that specify the transitions in the search space.
2. *Cost*. Definitions for a function specifying the cost (in a decision-theoretic sense) of a clause.
3. *Prune*. Definitions specifying clauses that can be admissibly removed from the search-space.

With some abuse of notation, it is possible to state the search procedure implicit in Steps 5–6 of Fig. 1. This is shown in Fig. 3.

The concept of enumerating clauses in a search procedure by repeated application of a refinement operator was introduced in Ref. [42]. The use of a user-specified cost function is a fundamental construct in statistical decision theory, and provides a general method of scoring descriptions. With an arbitrary cost function, it is usually not possible to devise automatically admissible strategies for pruning the search space. Thus it is also necessary to specify the *prune* statements for achieving this. One other construct that has proven useful is that of user-defined constraints. These specify the clauses that are unacceptable as hypotheses, but which cannot be removed admissibly. While this is most naturally specified by assigning an infinite cost to such clauses, it is sometimes more efficient to have separate constraint statements.

search($\perp, B, e$):

1. Let $S = \{\square\}, D = e, cost_{min} = cost(D, B)$
2. if $S = \{\}$ return D
3. Let $C \in S, S = S - \{C\}$
4. If $prune(C, B)$ then goto Step 2
5. If $cost(C, B) < cost_{min}$ then let $cost_{min} = cost(C, B), D = C$
6. Let $S = S \cup \{C_i \cdot \theta : C_i \in refine(C, B),\ \theta\ s.t.\ \square \succeq C_i \cdot \theta \succeq \perp\}$
7. Goto 2

Fig. 3. The search procedure using user-specified definitions for *refine*, *cost* and *prune*. Should lazy evaluation be permitted, then $\succeq$ becomes $\succeq_X$ and $\perp$ becomes $\perp_X$.

In this paper, when predicting numerical response variables, the mean-square-error of prediction (MSEP) on the training sample will be used as a measure of the cost of a clause performing such a prediction. Infinite cost will be assigned to clauses that do not compute the response variable. This is equivalent to specifying a constraint that disallows such clauses. The reader would note that by defining the cost function in the manner stated, we are unaffected by the fact that "negative" examples may not exist.

4. Relation to other work

The problem of the limitations in numerical capabilities of existing ILP systems has been addressed variously in the literature by either restricting the language of hypotheses to logical atoms [12], using built-in definitions for inequalities [3,33,35] or regression and numerical derivatives [13,14], transformations to propositional level [18] or constraint satisfaction problems [21,41], or using background knowledge for qualitative regression-like predicates [28]. The aims of this paper and the ideas developed here also bear a strong resemblance to the proposals made independently by Dzeroski in his doctoral dissertation [9] which describes the LAGRANGE system. This can be formulated as an ILP program with specific background predicate definitions for sines, cosines, multiplication, numerical differentiation and linear regression. Recent developments have also seen the concept of propositional regression trees being lifted to the first-order setting [17].

This paper contributes to this research in the following ways. First, the approach has sought to retain the generality of an ILP program. The technique of lazy evaluation is not specific to any particular predicate and allows arbitrary functions (statistical, numerical or even other propositional learners) to be used as background knowledge. To this extent, there has been no need to restrict the hypothesis language (as in Ref. [12]), use built-in predicates, single out regression-like predicates for special attention (as in Ref. [14]), or perform transformations. The resulting ILP program should in principle, be capable of learning directly theories of the form

reported in Ref. [8,21], or first-order definitions that achieve the aims of regression-trees [2] or model-trees [34]. It is more difficult to see how the theories obtained relate to the work in Ref. [41], although it is possible to achieve a form of clustering (see Ref. 3) making it somewhat similar in spirit to that work. Although the combined use of LAGRANGE and an ILP program is suggested in Ref. [9] it appears not to have been implemented. The results here can be viewed as providing evidence of how such hybrid methods work in practice.

Second, the experiments reported here provide a systematic assessment, in both controlled and realistic environments, of the numerical capabilities of an ILP program in comparison to some standard methods of quantitative analysis. The results provide some evidence to question any notion that the quantitative analysis by ILP programs is inherently inferior to other methods.

Finally, the use of lazy evaluation and user-defined search specification make contributions to the implementation of ILP systems. We note that the ability to specify a cost function provides a decision-theoretic interpretation of language restrictions like maximum "noise" allowed, constraints, etc. Further, an encoding of the refinement operator within the background knowledge can be seen the procedural equivalent of specifying the declarative language bias provided to programs like CLAUDIEN [36].

4.1. A note on programs used in experiments

The details described in Section 3 have been implemented in a Prolog program capable of emulating the algorithm in Fig. 1. The resulting program P-Progol (Version 2.3) is available on request from the first author. For convenience, in the rest of this paper we will refer to P-Progol (Version 2.3) as P-Progol. The pole-and-cart simulator and associated controller used in this paper was written by Professor C.A. Sammut. Readers interested in obtaining this program should contact him directly at the School of Computer Science and Engineering, University of New South Wales, Kensington, NSW 2052, Australia. Experimental results also tabulate the performance of the following propositional procedures that are more usually associated with quantitative analysis: (a) a stepwise regression procedure as implemented within the SPSS software package [30]; (b) a procedure that constructs "regression-trees" as implemented within the CART family of programs [2]; and (c) a backpropagation neural-network algorithm, implemented in the Stuttgart Neural Net Simulator (SNNS) [31].

5. Experiment 1: the pole-and-cart

The experiment concerns extracting constraints – in the form of equations – describing the linear and angular accelerations for a pole-and-cart system, using the variables describing the system state (see Fig. 4). The reader would note that the task here is not to construct a controller, but to extract the equations embodied in the pole-and-cart simulator. At this stage, we are also not concerned with the actual time for theory construction – the principal focus being a test of the ability to reconstruct the constraints.

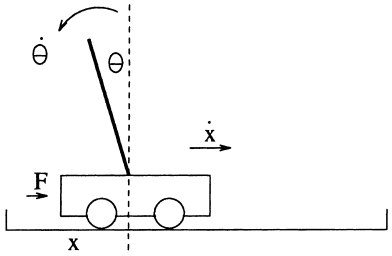

Fig. 4. The pole-and-cart problem. This refers to maintaining a pole balanced on a cart and keeping the cart within pre-specified track limits. The variables annotating the figure are: x, position of the centre of mass of the cart along the X axis; $\dot{x}$, linear velocity of the system; θ, angle of inclination of the pole; and $\dot{\theta}$, the angular velocity of the pole. Also calculable are the linear and angular accelerations $\ddot{x}$ and $\ddot{\theta}$. The mass of the pole is denoted by m and that of the cart by M. F refers to a fixed magnitude force that can be exerted on the cart to push it to the left or right ("bang-bang" control).

5.1. Experimental aims

From simulator records of a controller balancing a pole on a cart:

1. Determine if P-Progol, equipped with background definitions for arithmetic, statistical and numerical predicates, is capable of obtaining equations describing the linear and angular accelerations for the pole-and-cart system.
2. Compare the predictions made by any equations so obtained and those made by linear regression, regression-tree and neural-network methods on new data obtained under the same operating conditions.

5.2. Materials

5.2.1. Data

Data records available are tabulations of the values of specified numerical variables over a number of time-instants arising from repeated runs of a pole-and-cart simulator under the control of the BOXES algorithm [20]. Data are from 1000 runs of each of 9 pole-and-cart configurations arising from different forces and mass ratios of pole-and-cart. A run of the pole-and-cart simulator terminates when the simulation is unable to keep the pole upright, and the data are recorded at a prespecified sampling rate. Examples for analysis are obtained by randomly selecting one record from each run. This yields 1000 randomly selected examples for each configuration of pole-and-cart. Independence of each run of the pole-and-cart is assumed. There is no meaningful notion of negative examples for the constraints being learnt here, and each randomly selected example acts as a positive example for the ILP program.

5.2.2. Background knowledge for ILP

Complete Prolog listings of the background knowledge available to P-Progol are available in Ref. [43]. The contents can be summarised under the following categories:

1. *Simple arithmetic and trigonometry*. These include "built-in" definitions for $+, -, \times, /, ^{**}$ (power), sin, and cos.

2. *Inequalities*. The inequality $\leqslant$ is used to bound the maximum error allowed by an equation. With equations constructed by linear regression this is usually up to two standard deviations from the value predicted.
3. *Regression models*. The definition of multiple linear regression that minimises least-square error is included. Only those regression models that achieve a goodness-of-fit *F-test* probability of at least 0.01, and coefficient of multiple determination (that is, r^2) of at least 0.80 are deemed acceptable. These two parameters are typically used to determine if linear models obtained are acceptable [46], and their values, as adopted here, avoid poorly fitting models from being explored. Linear models with at most two independent variables are sufficient, as shown by the target model in Appendix B. Non-linear models are obtained naturally by the combined use of the power function (**) and linear regression.
4. *Numerical differentiation*. The use of a 5-point formula for calculating first-order derivatives of time-varying quantities [48] is explored when analysing the pole-and-cart data. The definition is constrained to obtain values of linear and angular accelerations from the corresponding velocities.
5. *Search specification*. The refinement operator used is the usual one defined in Ref. [24]. The cost function defined directs P-Progol to find concepts that minimise the mean square error arising from predictions on the training set of the response variable ($\ddot{x}$, $\ddot{\theta}$). For this cost function, the only pruning defined removes those clauses in P-Progol's hypothesis space that contain irrelevant additional literals (after computing the response variable), or those that could not possibly be compute the response variable within the constraints on the hypothesis language. A constraint specifies the straightforward requirement that clauses must contain an equation for the response variable along with error-bounds on the estimates made by any equations.

5.2.3. Attributes for propositional learners

For a fair comparison with propositional procedures, it is necessary that these procedures are also provided with attributes that encode the same background information. All attributes that can be constructed within the hypothesis language for P-Progol are provided to the propositional programs. A listing of these is available in Ref. [43].

5.3. Method

Appendix B contains a brief description of the simulator equations that act as target descriptions for experiments here. In these experiments, equations for linear and angular acceleration are constructed separately. When obtaining constraints for $\ddot{x}$, P-Progol has access to tabulated values of $x, \dot{x}, \theta, \dot{\theta}$, and $\ddot{\theta}$ for the examples chosen. Similarly, when obtaining constraints for $\ddot{\theta}$ the ILP program has access to values of $x, \dot{x}, \theta, \dot{\theta}$, and $\ddot{x}$. The following method was adopted to generate data, obtain and test any constraints:

1. 1000 runs are performed with each of the following values of force (F in N), mass of pole (m in kg), and mass of cart (M in kg): (i) $F = 0.0,\ m = 0.1,\ M = 1.0$; (ii) $F = 0.0,\ m = 1.0,\ M = 1.0$; (iii) $F = 0.0,\ m = 1.0,\ M = 0.1$; (iv) $F = +10.0,\ m = 0.1,\ M = 1.0$; (v) $F = +10.0,\ m = 1.0,\ M = 1.0$; (vi) $F = +10.0,\ m = 1.0,\ M = 0.1$; (vii) $F = \pm 10.0,\ m = 0.1,\ M = 1.0$; (viii) $F = \pm 10.0,\ m = 1.0,\ M =$

1.0; and (ix) $F = \pm 10.0,\ m = 1.0,\ M = 0.1$. On each run the values of $x, \dot{x}, \theta$, and $\dot{\theta}, \ddot{x}, \ddot{\theta}$ are tabulated once every 0.02 s.
2. For each configuration of F, m, M, a data record is selected at random. This yields 1000 data records for each configuration. The first 500 of these are used for "training" and the remainder are set aside to "test" any constraints obtained.
3. Using tabulated values for $x, \dot{x}, \theta, \dot{\theta}, \ddot{\theta}$, and the background knowledge obtain equations for $\ddot{x}$.
4. Using tabulated values for $x, \dot{x}, \theta, \dot{\theta}, \ddot{x}$, and the background knowledge obtain equations for $\ddot{\theta}$.
5. Record $\ddot{x}$ and $\ddot{\theta}$ values computed by these equations on the test set.
6. For each of $\ddot{x}$ and $\ddot{\theta}$, use training-set values for the attributes in Section 5.2.3 to (a) obtain a linear equation using stepwise linear regression; (b) obtain a regression tree; and (c) train the neural net. Record $\ddot{x}$ and $\ddot{\theta}$ values computed by each of these methods on the test set.
7. Compute the degree of agreement of each method to the actual values of $\ddot{x}$ and $\ddot{\theta}$ by computing the mean of the squared difference between actual values and those predicted by each method. In keeping with Ref. [40], this statistic is termed the "MSEP".

Parameter settings for the stepwise regression procedure control the inclusion and exclusion of variables in the equation. These parameters relate to the level of significance that has to be achieved above (or below) which a variable is retained in the model (or removed from the model). There is no prescribed setting for these parameters (termed *PIN* and *POUT*, respectively, in SPSS). We have adopted the procedure of setting $PIN, POUT$ to values that result in equations with no more than two independent variables on the training data, except when predicting $\ddot{x}$ with $F = \pm 10$ N. In this case, up to three independent variables are allowed, to enable the program to use values of F in the equation. This is in accordance with the restrictions provided to the ILP program.

The regression-tree procedure implemented within CART is equipped with validation procedures that enable it to estimate automatically the parameters specifying the tree-construction.

For the neural-network algorithm in SNNS, 10000 epochs were used with the "shuffle" option. In each epoch all training examples were presented. The net has one input unit for each attribute, four hidden units and one output unit was fully connected. There is no accepted method for determining the number of hidden units. The settings chosen were obtained based on the fact that they yielded the lowest error on the training data across possible settings ranging from 0 to 5 units in the hidden layer.

5.4. Experimental results and discussion

Figs. 5 and 6 tabulate values of MSEP summarising the difference between actual values of $\ddot{x}, \ddot{\theta}$ and those predicted by each method. The actual equations on which these calculations are based are in Appendix C.

The tabulations show that the MSEP from the ILP theory is usually lower than all programs other than regression. Fig. 7 tabulates a comparison of the MSEP of P-Progol against that of the propositional learners for the 18 errors tabulated in Figs. 5 and 6.

Algorithm	$F = 0$			$F = +10$			$F = \pm 10$		
	m/M			m/M			m/M		
	0.1	1.0	10.0	0.1	1.0	10.0	0.1	1.0	10.0
P-Progol	$9e-6$	$9e-6$	$2e-5$	$6e-5$	$1e-5$	$6e-4$	$4e-5$	$4e-4$	$2e-2$
Regression	$9e-6$	$9e-6$	$2e-5$	$6e-5$	$1e-5$	$6e-4$	$6e-5$	$1e-5$	$1e-2$
Regression tree	$3e-5$	$4e-4$	$3e-3$	$3e-5$	$7e-4$	$2e-2$	$5e-5$	$1e-3$	$3e-2$
Neural network	$2e-5$	$6e-4$	$5e-3$	$3e-5$	$1e-3$	$3e-2$	$8e-4$	$1e-2$	$1e-1$

Fig. 5. Mean square error of predicted values of $\ddot{x}$ on 500 test examples. The notation $ae-b$ stands for $a \times 10^{-b}$.

Algorithm	$F = 0$			$F = +10$			$F = \pm 10$		
	m/M			m/M			m/M		
	0.1	1.0	10.0	0.1	1.0	10.0	0.1	1.0	10.0
P-Progol –	$2e-3$	$1e-3$	$2e-3$	$2e-3$	$2e-3$	$1e-3$	$2e-3$	$2e-3$	$2e-3$
Regression	$2e-3$	$4e-4$	$8e-4$	$2e-3$	$2e-3$	$5e-2$	$1e-2$	$8e-3$	$7e-2$
Regression tree	$3e-3$	$5e-3$	$2e-2$	$3e-3$	$1e-2$	$4e-2$	$5e-3$	$1e-2$	$1e-1$
Neural network	$6e-3$	$1e-2$	$3e-2$	$5e-3$	$9e-3$	$1e-1$	$1e-1$	$1e-1$	$4e-1$

Fig. 6. Mean square error of predicted values of $\ddot{\theta}$ on 500 test examples. The notation $ae-b$ stands for $a \times 10^{-b}$.

Algorithm	MSEP of P-Progol		
	Better	Worse	Same
Regression	5	4	9
Regression tree	16	1	1
Neural network	17	1	0

Fig. 7. A comparison of the MSEPs on the pole-and-cart data. The terms "better", "worse" and "same" denote the cases that the MSEP of P-Progol is lower, higher, or the same (up to the precision shown in Figs. 5 and 6) as the corresponding propositional learner.

In general, we would expect the predictivity of P-Progol's theories to be at least comparable to those of linear regression given that the ILP program relies on regression definitions provided as background knowledge to construct its equations. Further, since by virtue of its search technique, P-Progol would do an "all-subsets" regression, it would seem to be surprising to find instances where the SPSS regression procedure (which implements "stepwise" regression) actually does perform better. Closer examination reveals that in 3 of the 4 cases that this occurs in Fig. 7, P-Progol has identified the correct target equation. This suggests its higher error to be an ar-

tifact of the test sample. To this extent, the entries in the first row of Fig. 7 are as expected.

On both data sets, the regression tree's predictions appear to be considerably worse than those made by P-Progol. By producing axis-parallel partitions and predicting mean values for each partition, the tree program is unable to capture the linear nature of the constraints involved. Definitive statements concerning the apparent poorer performance of the neural network are confounded by the absence of a principled technique for selecting the topology of the network. We can therefore do no more than state that for the particular topology used, the neural network's predictions are consistently worse than P-Progol's on the pole-and-cart data.

Besides a comparison of predictive accuracies, it is instructive to examine, where possible, the nature of the theories obtained by each algorithm. Appendix C shows that for the pole-and-cart problem, P-Progol constructs 24 equations corresponding to different physical parameter settings. Of these, the reader can verify that the correct linear model is identified on 22 occasions. Here, by "correct" we mean that the linear model has the same predictor variables as those in the target model described in Appendix B – we will examine the issue of correctly estimating the coefficients for these variables in greater detail below. In contrast, linear regression constructs 18 equations, of which 9 are correct. The difference in the number of equations highlights an important point of difference between the two programs when constructing theories for the case where $F = \pm 10$ N. Here, P-Progol constructs a pair of equations corresponding to the situations arising from $F = +10$ N, $F = -10$ N. The regression program attempts to explain both situations by using F as an independent variable in equations for $\ddot{x}$. Given the nature of their theories, we are not in a position to directly compare the output of the regression tree and neural network against the target model. On the pole-and-cart data, the tree program produces reasonably complex theories (some upto 200 nodes), and the latter's "theory" consisting of a listing of 25 floating-point numbers corresponding to the weights associated with each node in the network.

The availability of an automated controller (here the BOXES program) and a known target theory allows us to investigate further the nature of theories obtainable from P-Progol. In what follows, we restrict attention to a commonly used pole-and-cart configuration, namely $F = \pm 10$ N, $m = 0.1$ kg, $M = 1.0$ kg. Within this setting, we are in a position to examine empirically: (a) the convergence of coefficients in P-Progol's equations to the coefficients in the target theory; (b) bias in prediction and the estimation of the coefficients by P-Progol; and (c) extensions to the background knowledge to allow numerical differentiation by P-Progol.

We first examine how P-Progol's estimation of the target theory changes with increasing the number of training examples. Target theories for $\ddot{x}$ and $\ddot{\theta}$ are obtained from the equations of motion given in Appendix B. These are of the form $\ddot{x} = C_1 - M_1\ddot{\theta}\cos\theta + M_2\dot{\theta}^2\sin\theta$ and $\ddot{\theta} = C_2 + K_1\sin\theta - K_2\ddot{x}\cos\theta$. For the given set of physical parameters, namely $F = \pm 10$ N, $m = 0.1$ kg, $M = 1.0$ kg, the coefficients take the particular values: $C_1 = \pm 9.091$, $M_1 = M_2 = 0.045$, $C_2 = 0.000$, $K_1 = 14.7$, $K_2 = 1.50$ (the two values of C_1 result from $F = \pm 10$ N). Fig. 8 tabulates the progressive change in error of the P-Progol estimates from these expected values, averaged over the two values of F. The tabulation suggests that P-Progol's approximation of the target theory does improve as the number of training examples is increased.

Training Examples	Error in coefficients for $\ddot{x}$	Error in coefficients for $\ddot{\theta}$
100	–	0.803
250	0.002	0.587
500	0.002	0.876
1000	0.002	0.350
1500	0.001	0.141

Fig. 8. Error of coefficients estimated in the linear model obtained by P-Progol for the pole-and-cart data. For a given number of training examples, this error is the average of the total absolute deviation of estimates from expected values (from the target model) for $F = +10$ and $F = -10$. Estimates and expected values are recorded up to three significant figures. An entry of "–" denotes that the target model was not identified by P-Progol.

We now turn our attention to the question of bias in P-Progol's predictions and estimates of the coefficients. For the physical settings under consideration, Fig. 9 tabulates the frequencies of positive and negative residuals arising from over and underestimates of $\ddot{x}$ and $\ddot{\theta}$ on the 500 test examples. Also tabulated are the means of the predicted and actual values for each variable. The entries suggest that P-Progol's predictions appear to be largely unbiased.

Consider now any bias in the estimation of coefficients that appear in equations for the dependent variables. Operating under the control of a two-sided force provides us with two estimates for each coefficient – one for which F is positive (here $+10$ N) and the other when it is negative (-10 N). Calling these two estimates "duplicates", [1] and repeatedly performing the experiment of learning equations for $\ddot{x}$ and $\ddot{\theta}$ allows us to record the number of occasions that (a) both duplicates overestimate a coefficient; (b) both duplicates underestimate a given coefficient; (c) the F-positive duplicate overestimates a coefficient while the F-negative underestimates; (d) the F-positive duplicate underestimates a coefficient while the F-negative overestimates; (e) one duplicate under or over-estimates a coefficient while the other is exact (up to some degree of precision); and (f) both duplicates estimate a coefficient exactly (again, up to some degree of precision). Severe discrepancies between the tallies in (a) and (b) would suggest that P-Progol's estimation of that coefficient was biased. Discrepancies in the tallies (c) and (d) would suggest bias in the simulator. Tallies of (e) and (f) are not of particular interest here.

Fig. 10 tabulates these numbers over 10 repeats of obtaining equations for $\ddot{x}$ and $\ddot{\theta}$ from independent training sets of 500 examples each. The 6 cases (a)–(f) in the tabulation correspond to the situations described above, and the coefficients $C_1, \ldots, K_2$ are as before. The comparison of estimates and expected values proceeds to 3 significant figures. It is evident from this figure that for 4 of the 6 coefficients, there is no evidence of any bias. Of the remaining two, the evidence for bias in estimates of M_1 does not appear to be substantial. Some degree of uncertainty does however rest on the estimates of M_2, and suggests further investigation.

[1] This term and the analysis following are due to Donald Michie.

Dependent variable	Frequency		Mean	
	+ residuals	− residuals	Actual	Predicted
$\ddot{x}$	0.46	0.52	−0.195	−0.196
$\ddot{\theta}$	0.54	0.46	0.309	0.310

Fig. 9. Examining residuals for bias in P-Progol's prediction of the dependent variable on the pole-and-cart data.

Case	Predicting $\ddot{x}$			Predicting $\ddot{\theta}$		
	C_1	M_1	M_2	C_2	K_1	K_2
(a)	0	0	5	1	0	3
(b)	0	3	0	1	0	3
(c)	0	0	2	5	0	1
(d)	0	0	0	3	0	1
(e)	0	5	3	0	2	2
(f)	10	2	0	0	8	0

Fig. 10. Occurrences of cases arising from duplicate estimates over, under or exactly estimating a given coefficient in the linear models describing the pole-and-cart data. The cases (a)–(f) are as described in the text. As before, estimates and expected values are recorded up to three significant figures.

In the experiments undertaken, the reader would have noticed that the target equations for $\ddot{x}$ require the values of $\ddot{\theta}$ and vice-versa and that P-Progol obtained these values directly from the tabulated records. It is of interest to examine whether the ILP program could achieve comparable results by calculating the derivatives required by using background knowledge definitions for numerical differentiation. This would allow the ILP program to emulate other, more special purpose programs like LAGRANGE [8]. We close this discussion with a demonstration that such behaviour is possible by extracting constraints for $\ddot{x}$ and $\ddot{\theta}$ using $x, \dot{x}, \theta$, and $\dot{\theta}$ in conjunction with a well-known 5-point formula for numerical differentiation. Examples are time-indexed to allow such a calculation. The physical parameters remain as before, namely $F = \pm 10$ N, $m = 0.1$ kg, $M = 1.0$ kg, and the corresponding mean-square-error on the test sample is in Fig. 11. The errors can be seen to be comparable to the corresponding ones obtained earlier in Figs. 5 and 6. However, it is important to note here that the 5-point formula for numerical derivatives are not calculable for examples that are too close to the edges of a run. Prediction of values is thus possible for only a subset of the data – an issue that is not peculiar to the use of P-Progol, but one that arises from the particular numerical method employed to obtain derivatives.

5.5. Conclusion

The experiments in this section have concentrated on the relatively well-defined world of the pole-and-cart. The conclusions to be drawn from this are straightfor-

Variable	Mean square error
$\ddot{x}$	$3e-5$
$\ddot{\theta}$	$2e-3$

Fig. 11. Mean square error of P-Progol predictions for $\ddot{x}$ and $\ddot{\theta}$ on test examples with programs for numerical differentiation provided as background knowledge. The notation $ae-b$ stands for $a \times 10^{-b}$. Note that the errors reported are calculated only on those examples in the test-set for which numerical derivatives are defined.

ward, namely: (1) when the data are fully predictable from simple linear equations, linear modelling does as well or better than methods unable to express such models; and (2) regression used within an ILP harness to exhaust the subsets of possibly predictive independent variables does better than following the greedy strategy of "stepwise" regression. There are thus, no surprises at all for data analysts. The experiments do however serve to illustrate the possibility of using statistical and numerical predicates within an ILP program. We now consider a case where there are no known models for predicting the data. This provides a sterner test of the capabilities of the ILP program.

6. Experiment 2: mutagenesis

There are two broad stages in rational drug design [39]. The first involves the identification of one or more compounds – known as leads – with some suitable biological activity. This activity is obtained from historical records, or chemical assays. The second stage involves optimising the activity of these leads. Typically, the medicinal chemist has access to the 2- and 3-dimensional structure of the possible leads, along with their activities. Empirical Structure-Activity Relationships – or SARs – are concerned with obtaining accurate descriptions that describe levels of biological activity in terms of molecular structure. These descriptions can then be used to direct the synthesis of new compounds, possibly culminating in a new drug. The SAR problem here is taken from the chemical literature as reported by [6]. The authors are concerned with obtaining SARs describing mutagenicity in nitroaromatic compounds. These compounds occur in automobile exhaust fumes and are also common intermediates in the synthesis of many thousands of industrial compounds [6]. Highly mutagenic nitroaromatics have been found to be carcinogenic, and it is of considerable interest to the pharmaceutical industry to determine which molecular features result in compounds having mutagenic activity.

Since its introduction in Ref. [45], the problem has become an important test-bed for ILP programs. Most of this research has however concentrated in obtaining rules for classifying the compounds into one of several categories, although the original problem is concerned with the quantitative task of predicting actual mutagenic activity. In Ref. [6] the authors list the activity of 230 compounds, obtained from a procedure known as the Ames test. They further identify this set as being composed of two disparate groups of 188 and 42 compounds respectively. The first group is adequately explained by linear modelling using five specifically selected predictor vari-

ables. The remaining 42 compounds however form an "outlier" group for which no explanatory model has been proposed by the chemists. It is this subset of compounds that are of particular interest to us. They provide the opportunity of examining whether the first-order capabilities of a program like P-Progol provide the edge required to find explanatory models. This capability allows P-Progol to include relational descriptions in terms of molecular structure, thus going beyond the routine use of propositional algorithms like regression. Recent work on this subset of data [44] examines augmenting the independent variables used by linear regression with new "features" constructed by an ILP program. In some sense, the technique used in this paper can be seen as a dual to that work (in which the results of an ILP program are used by linear regression). The advantage of the technique adopted here, as seen below, is that it allows a uniform treatment of mixed class types (that is, both numerical and nominal).

6.1. Experimental aims

For the 42 nitroaromatic molecules not explained by chemists:

1. Determine if P-Progol, equipped with background definitions for describing molecular structure, the expert selected predictor variables, and statistical predicates, is capable of obtain an explanation for the mutagenic activity of the molecules.
2. Compare the predictions made by the P-Progol theory and those made by regression, regression-tree and neural-network methods using the expert selected predictor variables only.

We note here that a chemical evaluation of any theory constructed by P-Progol is beyond the scope of this paper.

6.2. Materials

6.2.1. Data

Data are available for the mutagenic activity of 42 compounds. Of these, 20 compounds have recordable levels of activity. The activity of the remaining 22 compounds is below the minimum levels measurable. These have been marked simply as "very low". This poses special problems for programs that are incapable of dealing with mixed data types. While there is no natural notion of "negative" examples for the 20 compounds for which numeric activity levels are available, it is possible to view them as acting as negative examples when learning rules for "very low" activity (that is, for the remaining 22 compounds).

6.2.2. Background knowledge for ILP

Besides the 5 predictor variables devised by the chemists (see Section 6.2.3), P-Progol has the following additional information as background knowledge (complete Prolog listings of these are available in Ref. [43]):

1. *Molecular structure*. These are primitive structural descriptions in terms of the atom and bond structures in each molecule, along with associated information concerning atom/bond types. They are represented by a pair of predicates *atm/5* and *bond/4* (as in Ref. [15]), and are obtained automatically from a molecular modelling package.

2. *Inequalites*. The inequality $\leqslant$ is used to bound values attained by numeric predicator variables, partial charges on atoms, and the maximum error allowed by a regression equation.
3. *Regression models*. The definition of multiple linear regression that inimises least-square error is included. As before, only those regression models that achieve a goodness-of-fit *F-test* probability of at least 0.01, and coefficient of multiple determination (that is, r^2) of at least 0.80 are deemed acceptable. Further, models are again restricted to those with at most two independent variables.
4. *Expected values*. A function that computes (lazily) the expected value of the activity of a set of compounds.
5. *Search specification*. For efficiency, we define a refinement operator that constrains atom or bond descriptions to appear at most once in a hypothesised clause. P-Progol does not have a difficulty with mixed data types, and the cost function assigns (a) mean-square-error as the cost for clauses computing a numeric value for activity; and (b) lack of "compression" of the examples as the cost for clauses classifying compounds as having "very low" activity. No pruning was specified.

6.2.3. Expert-selected attributes

The following attributes have been used in Ref. [6] to obtain SARs:

Attribute	Description
ϵ_{LUMO}	Energy level of lowest unoccupied molecular orbital in compound
$\log(P)$	Hydrophobicity of compound
I_1	Logical attribute: 1 if compound contains three or more benzyl rings
I_a	Logical attribute: 1 if compound is an acenthrylene
$\log_{10}(10^{(\log(P)-5.48)}+1)$	Attribute constructed by chemists (called Hansch attribute)

6.3. Method

The small number of compounds (42, of which only 20 have numeric values) forces any estimates of the predictive power of models obtained to be necessarily speculative. We adopt the following experimental design:

1. Using the background knowledge described in Section 6.2.2 and the expert-selected attributes in Section 6.2.3 construct P-Progol theories explaining the activity of the 20 compounds with numeric activity, and the 22 compounds classified as "very low".
2. Using the expert selected attributes described in Section 6.2.3 for the 20 compounds with numeric activity: (a) obtain a linear equation using stepwise linear regression; (b) obtain a regression tree; and (c) train the neural net.
3. Find estimates of the MSEP by obtaining leave-one-out predictions [47] by all algorithms for the 20 compounds. Two complications can arise. First, P-Progol's theory may be overly specific. This may result in the activity of some compounds not being predicted. Second, more than one rule may be applicable, resulting in

multiple predictions of activity. For the latter, we adopt the convention of using the first rule that is applicable. The same effect is achieved by using *refine* definitions that include Prolog cuts ("!") at the end of each clause. For the former, we tabulate MSEP values obtained by (a) ignoring non-predicted compounds; and (b) assigning the mean activity of the training set to such compounds (this acts as a "default" rule).

The reader will note that comparative statistics are only obtainable on the subset of compounds with numeric activity values. The reader could correctly question the value of obtaining a theory using stepwise linear regression, as this has already been rejected by the chemists. We do so here only for completeness. For this, the SPSS regression parameters *PIN*, *POUT* are progressively relaxed until an equation is obtained. Parameter settings for the regression-tree and neural-network were obtained in the manner described earlier – namely, automatic optimisation (regression-tree) and manual experimentation with a range of topologies (neural-network).

6.4. Experimental results and discussion

Fig. 12 tabulates values of MSEP summarising the difference between actual values of mutagenic activity and those predicted by each algorithm. P-Progol (1) refers to the error when non-predicted compounds were ignored. There were five such compounds. P-Progol (2) refers to the error when these compounds were assigned mean activity values of the training set. 'Default" refers to the strategy of prediction being the mean activity of the training set. We note here that the internal optimiser within the regression-tree program is unable to find a good tree for the 20 compounds.

The tabulation in Fig. 12 shows that when it predicts a value, P-Progol's model has the lowest MSEP. However, this edge appears to be reduced to no better than mean-value prediction once augmented by the default rule to enable prediction of such compounds. Leave-one-out comparisons of the MSEP values do not however bring out differences in the explanatory power provided by the methods. A measure of the association of predicted and actual values is obtained by the correlation between these two quantities using complete theories for the 20 compounds. This comparison is shown in Fig. 13. Rank correlation estimates are provided as they make no assumptions of normality. Given the small size of the sample, we do not perform any

Algorithm	MSEP
P-Progol (1)	1.8
P-Progol (2)	2.2
Regression tree	–
Regression	2.9
Neural network	3.1
Default	2.2

Fig. 12. Mean square errors of prediction on the 20 compounds with recorded activity values. The estimates are from leave-one-out cross-validation.

Algorithm	Correlation
P-Progol (1)	0.84
P-Progol (2)	0.71
Regression tree	–
Regression	0.65
Neural network	-0.2
Default	–

Fig. 13. Spearman's rank correlation estimates of true activity to values predicted by the theories. P-Progol (1) and (2) are as earlier. The regression-tree and default strategy predict the same activity for all compounds, making estimates of correlation meaningless.

tests of significance. These results therefore can only be taken as providing evidence for a further investigation, but they do serve to highlight the inadequacy of using the default rule for explanation. We note in passing that the rank correlations in Fig. 13 are similar to those reported in Ref. [44] for the same subset of 20 compounds (there a value of 0.64 is obtained).

The relatively poor performance of the neural network and the lack of a model from the regression tree have to accompanied by the caveat that better results may be possible with more experimentation with topological parameters (for the network) or language restrictions (for the tree). We do not pursue this further as such manipulations would then have to performed with other algorithms as well, which is beyond the scope of this study. That logical structuring using molecular structure aids predicitivity appears to be supported by the fact that P-Progol has lower error rates than programs unable to use such information. We also note that the models by P-Progol's were required to achieve a goodness-of-fit *F-test* probability of at least 0.01. No such models were available using regression only, confirming

If Molecule A has:
a Lumo value L1 of at most -1.35, and
a carbon atom C in a 6-membered aromatic ring with nitrogens around, and
a LogP value L2
then: its expected activity is -2.92 × L1 + 1.499 × L2 - 12.72

Else If Molecule A has:
a carbonyl carbon C with partial charge at most 0.608,
C is connected via a single bond to some atom D
then: its expected activity is 1.38

Else If Molecule A has:
a hydroxyl oxygen (i.e. the structure $H - O - < C >= O$)
then: its expected activity is very low

Fig. 14. P-Progol's theory for the 42 compounds. The theory shown is an English translation of the Prolog clauses found by P-Progol. These clauses leave as outliers 5 of the 20 compounds with numeric values, and 16 of the 22 compounds with "very low" activity.

the chemists opinion that no good models were directly obtainable from regression.

As in the case of the pole-and-cart, it is instructive to examine the actual theory constructed by P-Progol. Fig. 14 shows a text translation of this theory. It is evident that explanation is achieved by a combination of logical and statistical descriptors. The former are concerned with identification of chemical substructures, or numeric comparisons. The latter deal with linear models, or calculations of expected activity levels. Clauses use these descriptors to predict either a numeric or nominal value. The result therefore, is more sophisticated than piecewise linear modelling. Such a treatment of mixed data types illustrates well the type of analysis that can be achieved.

6.5. Conclusion

We are now in a position to build on the conclusions reached earlier from experiments with the pole-and-cart. The results here suggest: (1) when the data are known not to be predictable from simple linear models, an ILP program can achieve explanations where none are possible from (expert-guided) regression; and (2) an ILP program can naturally represent and analyse mixed data types, thus making it possible to achieve a combination of classification and numeric prediction within the one framework. Further, there appears to be some evidence that the predictivity of the ILP theory appears to be better than that of quantitative analytical methods like regression-trees and neural networks. However this requires further rigorous experimentation.

7. Concluding remarks

By adopting logic programs as its basic representation formalism, an ILP program can use and construct clauses in a Turing-equivalent language. In principle therefore, there is no restriction to the functions that can be used as background knowledge or learnt by such a program.

This paper has described two implementation extensions that allow an ILP program to exploit more fully the power afforded by the theoretical framework of ILP. We have then sought to demonstrate empirically how a program equipped with these changes could go some way towards redressing a perceived limitation of existing ILP programs, namely, the analysis of numerical data. The results from the experiments reported provide evidence that the analytical capabilities of an ILP program are not inferior to traditional quantitative methods like regression and neural-networks. The richer language and comprehensibility of ILP theories that follow naturally from their use of first-order logic, are of course, retained. We should emphasise here that the apparently confrontational nature of the experimental methodology is illusory as lazy evaluation allows any propositional algorithm to be used as background knowledge to an ILP program. This has not been exploited fully by the experiments in this paper, nor has the possibility of obtaining rules evaluation of which requires a general constraint-solver. The latter leads to the promising area of learning constraint logic programs. To this extent, we take the results here as providing further incentive for investigating the use of general-purpose ILP programs for the quantitative analysis of scientific and engineering data.

Acknowledgements

This research was partly supported by the EPSRC project 'Experimental Application and Development of ILP'. Ashwin Srinivasan currently holds a Nuffield Trust Research Fellowship at Green College, Oxford. At the time of conducting these experiments, Rui Camacho was supervised initially by Stephen Muggleton, and later by Pavel Brazdil. Support for Rui Camacho was provided by Universidade do Porto and JNICT (Junta Nacional de Investigação Científica e Tecnológica). Thanks are due to Donald Michie for his invaluable guidance and advice and to Stephen Muggleton and Claude Sammut for patient discussions on the experiments here.

Appendix A. Examples of implementation changes

This section gives some examples of the implementation, within P-Progol (version 2.3), of two changes proposed in Section 3 namely, lazy evaluation and user-definable search functions.

A.1. Lazy evaluation

We demonstrate here the construction of a regression line between a pair of points. Suppose $E^+ = \{p(0,1) \leftarrow, p(8,4) \leftarrow\}$, $E^- = \emptyset$, and B contains a correct definition for computing least-squares estimates for the slope (M) and intercept (K) of line drawn through a set of points. The target therefore, is to construct a clause of the form $p(X,Y) \leftarrow line(X,Y,m,k)$, where m,k are some specific values for M,K, with the first two arguments of *line* are "input", and the remainder are constants to be computed lazily.

In executing the lazy evaluation procedure described in Fig. 2, the following steps are followed:

lazyeval(l, C, h, B, E^+, E^-)

1. Here $l = line(X,Y,\exists M,\exists K), C = p(X,Y)$, $E^+ = \{p(0,1), p(8,4)\}, E^- = \emptyset$, B, h are given.
2. Then $\theta_p^X = \{X/0, X/8\}$, $\theta_n^X = \emptyset$ and $\theta_p^Y = \{Y/1, Y/4\}$, $\theta_n^Y = \emptyset$.
3. Call $\mathscr{LINE}([\theta_p^X, \theta_n^X], [\theta_p^Y, \theta_n^Y], M, K)$.
4. Background B contains definition $\mathscr{LINE}/4$ that returns (within h resolution steps) the answer substitution $\{M/0.375, K/1\}$.

Search now proceeds with the clause $p(X,Y) \leftarrow line(X,Y,0.375,1)$.

A.2. User-defined search

We give examples of the the three different search operations described in Section 3. Consider for example, the task undertaken in Ref. [27]. There, the task set for an earlier version of P-Progol is to find within an organic molecule, conjunction functional classes – like hydrogen donors, hydrogen acceptors, and zinc-binding sites – and the 3-dimensional distances between such classes. In Ref. [27] the authors use a rather circuitous method of specifying this requirement. The following *refine* clauses within P-Progol (version 2.3) achieve the same effect more directly (we do not show definitions for auxiliary predicates like *member* and *dist*).

```
refine(false,F active(A)).

refine(active(A),Clause):-
        member(Pred1,[hacc(A,B),hdonor(A,B),zincsite(A,B)]),
        member(Pred2,[hacc(A,C),hdonor(A,C),zincsite(A,C)]),
        member(Pred3,[hacc(A,D),hdonor(A,D),zincsite(A,D)]),
        member(Pred4,[hacc(A,E),hdonor(A,E),zincsite(A,E)]),
        Clause = (active(A):-
                    Pred1,
                    Pred2,
                    dist(A,B,C,D1,E1),
                    Pred3,
                    dist(A,B,D,D2,E2),
                    dist(A,C,D,D3,E3),
                    Pred4,
                    dist(A,B,E,D4,E4),
                    dist(A,C,E,D5,E5),
                    dist(A,D,E,D6,E6)).
```

Cost specification takes the form of definition of a 3-place predicate. The one extra argument is an efficiency concern that provides some pre-computed statistics of the clause (like number of positive and negative examples derivable and clause length). Here are the cost functions used in the experiments in this paper – again without detail of intermediate predicates.

```
% cost is mean-square-error for numeric calculations
cost((Head:-Body),Label,Cost):-
        mse(Head,Body,Cost),
        Cost\= undef,!.

% otherwise clauses compressing fewer examples are costlier
cost(_,[P,N,L],Cost):-
        P > L, N = 0, !,
        Cost is -P.

cost(_,_,inf).
```

A typical use of pruning statements may be to remove redundant literals in a search. A case in point are implication relationships arising from inequalities. Here is an example resulting from addition of a literal performing an inequality test. In the definition below, *in(L1,L2)* is a predicate that checks if a literal $L1$ is in a conjunction $L2$, and *add(L1,L2,L3)* adds a literal $L1$ to the end of a sequence of conjoined literals $L2$ to give $L3$.

```
prune((Head:-Body)):-
        add((X =< N2),L,Body),
        in((Y =< N1),L),
        X = = Y,
        N1 =< N2.
```

Appendix B. The pole-and-cart model

Exact equations of motion for the pole-and-cart system are available in the literature, and we do not include their derivation here. For this, the reader is referred to the treatment in Ref. [4], pp. 703–710. Instead, we merely reproduce the relevant equations here. The state of the system is given by θ: the angle of the pole from the upright position; $\dot{\theta}$: angular velocity of the pole; x: horizontal position of the cart's center; and $\dot{x}$: the velocity of the cart. Given a force on the cart of fixed magnitude F, and a pole of length l, cart and pole masses of M and m, the equations of motion are:

$$(M+m)\ddot{x} + \frac{1}{2}ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = F,$$

$$\ddot{x}\cos\theta + \frac{2}{3}l\ddot{\theta} = g\sin\theta.$$

These can be re-arranged to give constraints for $\ddot{x}$ and $\ddot{\theta}$:

$$\ddot{x} = \frac{F - \frac{1}{2}ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta)}{M+m},$$

$$\ddot{\theta} = \frac{g\sin\theta - \ddot{x}\cos\theta}{\frac{2}{3}l}.$$

Appendix C. A selection of theories obtained from P-Progol

C.1. Pole-and-cart

Actual Prolog definitions found by P-Progol are only of interest for syntactic reasons, and are presented for one configuration only. A mathematical representation of all equations obtained follow, and can be used to understand the Prolog representation. In constructing these clauses, P-Progol treated *lin_regress2* as a lazily-evaluated literal, and the constants that appear are a result of the mplementation described earlier. Thus, the literal *lin_regress2(G,I,K,14.667,−1.560,0.583,0.044,L)* encodes the regression line $G = 14.667 \times I - 1.56 \times K + 0.583$, with standard deviation $s = 0.044$ and standard residual L.

```
% F = +/- 10, m = 0.1, M = 1.0
accel(A,B,C,D,E,F,f(10.000),G,H) :-
          sin(A,B,E,I), cos(A,B,E,J), mult(A,B,J,H,K),
          lin_regress2(G,I,K,14.667,−1.560,0.583,0.044,L),lte(L,2.000).
accel(A,B,C,D,E,F,f(-10.000),G,H) :-
          sin(A,B,E,I), cos(A,B,E,J), mult(A,B,J,H,K),
          lin_regress2(G,I,K,14.686,−1.565,−0.633,0.040,L),lte(L,2.000).

accel(A,B,C,D,E,F,f(10.000),G,H) :-
          sin(A,B,E,I), cos(A,B,E,J), mult(A,B,J,G,K), pow(A,B,F,2,L),
```

```
        mult(A,B,I,L,M),lin_regress2(H,M,K,0.046,−0.046,9.088,
        0.003,N),lte(N,2.000).
accel(A,B,C,D,E,F,f(-10.000),G,H) :-
        sin(A,B,E,I), cos(A,B,E,J), mult(A,B,J,G,K), pow(A,B,F,2,L),
        mult(A,B,I,L,M),lin_regress2(H,M,K,0.047,−0.046,−9.090,
        0.003,N),lte(N,2.000).
```

C.2. Mutagenesis

The complete theory in Prolog form is shown below. The text translation appearing in Fig. 15 is of the non-ground clauses below. Both *lin_regress2*and *expected_value4* are lazily evaluated. The literal *expected_value(B,1.38,0.6144,F)*states that the set of values B have expected value 1.38 with standard deviation $s = 0.6144$, and standard residual F.

```
active(A,B) :-
      lumo(A,C), lteq(C,-1.35),
      logp(A,D), atm(A,E,c,22,F), bond(A,E,G,7),
      lin_regress2(B,C,D,-2.902,1.499,-12.72,0.6289,H), lte(H,2).
```

F	m/M	Target equation	Equation obtained
0	0.1	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = -0.002 + 12.644sin\theta - 4.543\ddot{x}cos\theta$
		$\ddot{x} = 0.0 - 0.045\ddot{\theta}cos\theta + 0.045\dot{\theta}^2sin\theta$	$\ddot{x} = 0.0 - 0.045\ddot{\theta}cos\theta + 0.043\dot{\theta}^2sin\theta$
	1.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = -0.0003 - 4.099\ddot{x}cos\theta + 1.067\dot{\theta}^2sin\theta$
		$\ddot{x} = 0.0 - 0.25\ddot{\theta}cos\theta + 0.25\dot{\theta}^2sin\theta$	$\ddot{x} = 0.0 - 0.25\ddot{\theta}cos\theta + 0.247\dot{\theta}^2sin\theta$
	10.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = 0.0 + 13.182sin\theta - 1.581\ddot{x}cos\theta$
		$\ddot{x} = 0.0 - 0.45\ddot{\theta}cos\theta + 0.45\dot{\theta}^2sin\theta$	$\ddot{x} = 0.0 - 0.455\ddot{\theta}cos\theta + 0.457\dot{\theta}^2sin\theta$
+10	0.1	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = 0.253 + 14.68sin\theta - 1.526\ddot{x}cos\theta$
		$\ddot{x} = 9.09 - 0.045\ddot{\theta}cos\theta + 0.045\dot{\theta}^2sin\theta$	$\ddot{x} = 9.091 - 0.045\ddot{\theta}cos\theta + 0.045\dot{\theta}^2sin\theta$
	1.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = 0.041 + 14.647sin\theta - 1.510\ddot{x}cos\theta$
		$\ddot{x} = 5.0 - 0.25\ddot{\theta}cos\theta + 0.25\dot{\theta}^2sin\theta$	$\ddot{x} = 5.0 - 0.25\ddot{\theta}cos\theta + 0.251\dot{\theta}^2sin\theta$
	10.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta$	$\ddot{\theta} = 0.0004 + 14.692sin\theta - 1.5\ddot{x}cos\theta$
		$\ddot{x} = 9.09 - 0.45\ddot{\theta}cos\theta + 0.45\dot{\theta}^2sin\theta$	$\ddot{x} = 9.094 - 0.454\ddot{\theta}cos\theta + 0.453\dot{\theta}^2sin\theta$
±10	0.1	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = +10)$	$\ddot{\theta} = 0.583 + 14.667sin\theta - 1.56\ddot{x}cos\theta$
		$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = -10)$	$\ddot{\theta} = -0.633 + 14.686sin\theta - 1.565\ddot{x}cos\theta$
		$\ddot{x} = 9.09 - 0.045\ddot{\theta}cos\theta + 0.045\dot{\theta}^2sin\theta \ \ (F = +10)$	$\ddot{x} = 9.088 - 0.046\ddot{\theta}cos\theta + 0.046\dot{\theta}^2sin\theta$
		$\ddot{x} = -9.09 - 0.045\ddot{\theta}cos\theta + 0.045\dot{\theta}^2sin\theta \ \ (F = -10)$	$\ddot{x} = -9.09 - 0.046\ddot{\theta}cos\theta + 0.047\dot{\theta}^2sin\theta$
	1.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = +10)$	$\ddot{\theta} = 0.556 + 14.338sin\theta - 1.57\ddot{x}cos\theta$
		$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = -10)$	$\ddot{\theta} = -0.574 + 14.314sin\theta - 1.573\ddot{x}cos\theta$
		$\ddot{x} = 5.0 - 0.25\ddot{\theta}cos\theta + 0.25\dot{\theta}^2sin\theta \ \ (F = +10)$	$\ddot{x} = 5.103 - 0.016\ddot{\theta}cos\theta + 0.241\dot{\theta}^2sin\theta$
		$\ddot{x} = -5.0 - 0.25\ddot{\theta}cos\theta + 0.25\dot{\theta}^2sin\theta \ \ (F = -10)$	$\ddot{x} = -5.001 - 0.250\ddot{\theta}cos\theta + 0.249\dot{\theta}^2sin\theta$
	10.0	$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = +10)$	$\ddot{\theta} = 0.075 + 14.639sin\theta - 1.503\ddot{x}cos\theta$
		$\ddot{\theta} = 0.0 + 14.7sin\theta - 1.5\ddot{x}cos\theta \ \ (F = -10)$	$\ddot{\theta} = -0.044 + 14.634sin\theta - 1.502\ddot{x}cos\theta$
		$\ddot{x} = 9.09 - 0.45\ddot{\theta}cos\theta + 0.45\dot{\theta}^2sin\theta \ \ (F = +10)$	$\ddot{x} = 9.096 - 0.454\ddot{\theta}cos\theta + 0.451\dot{\theta}^2sin\theta$
		$\ddot{x} = -9.09 - 0.45\ddot{\theta}cos\theta + 0.45\dot{\theta}^2sin\theta \ \ (F = -10)$	$\ddot{x} = -9.096 - 0.454\ddot{\theta}cos\theta + 0.453\dot{\theta}^2sin\theta$

Fig. 15. Descriptions of equations obtained by P-Progol on the pole-and-cart data.

```
active(A,B) :-
      atm(A,C,c,14,D), lteq(D,0.608), bond(A,C,E,1),
      expected_value(B,1.38,0.6144,F), lte(F,2).
active(A,vlow) :- atm(A,B,o,45,C).
active(e10,vlow).  active(e11,vlow).  active(e6,vlow).
active(e16,vlow).  active(e14,vlow).  active(e3,vlow).
active(e18,vlow).  active(e24,vlow).  active(e21,vlow).
active(e9,vlow).  active(e13,vlow).  active(e12,vlow).
active(e4,vlow).  active(e7,vlow).  active(e8,vlow).
active(e5,vlow).
```

References

[1] I. Bratko, M.Grobelnik, Inductive learning applied to program construction and verification, in: Proceedings of the Third International Workshop on Inductive Logic Programming. Available as Technical Report IJS-DP-6707, J. Stefan Inst., Ljubljana, Slovenia, 1993, pp. 279–292.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression-trees, Wadsworth, Belmont, 1984.

[3] R. Camacho, Learning stage transition rules with Indlog, in: S. Wrobel, (ed.), Proceedings of the Fourth International Inductive Logic Programming Workshop, Gesellschaft fur Mathematik und Datenverarbeitung MBH, GMD-Studien Nr 237, 1994.

[4] R.H. Cannon, Dynamics of Physical Systems, McGraw-Hill, New York, 1967.

[5] European Commission, ESPRIT Long term research project 20237 – ILP II, Technical Annex I, European Commission, Directorate General, Industry, Brussels, 1996.

[6] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Schusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds, correlation with molecular orbital energies and hydrophobicity, J. Medicinal Chem. 34 (2) (1975) 786–797.

[7] B. Dolsak, S. Muggleton, The application of inductive logic programming to finite element mesh design, in: S. Muggleton, (Ed.), Inductive Logic Programming. Academic Press, London, 1992, pp. 453–472.

[8] S. Džeroski, L. Todorovski, Discovering dynamics: From inductive logic programming to machine discovery, J. Intell. Inform. Sys. 4 (1995) 89–108.

[9] S. Dzeroski, Numerical Constraints and Learnability in Inductive Logic Programming, University of Ljubljana, (PhD. Thesis), Ljubljana, 1995.

[10] S. Dzeroski, L. Dehaspe, B. Ruck, W. Walley, Classification of river water quality data using machine learning, in: Proceedings of the Fifth International Conference on the Development and Application of Computer Techniques Environmental Studies, 1994.

[11] C. Feng, Inducing temporal fault dignostic rules from a qualitative model, in: S. Muggleton, (Ed.), Inductive Logic Programming, Academic Press, London, 1992, pp. 473–486.

[12] C.D. Page Jr., A.M. Frisch, Generalization and learnability: A study of constrained atoms, in: S. Muggleton, (Ed.), Inductive Logic Programming, Academic Press, London, 1992, pp. 29–56.

[13] A. Karalic, Relational regression: first steps, Technical report ijs-dp-7001, J. Stefan Inst., Ljubljana, Yugoslavia, 1994.

[14] A. Karalic, I. Bratko, First-order regression (special issue on ILP), Machine Learning Journal 26 (1997) 147–176.

[15] R.D. King, S.H. Muggleton, A. Srinivasan, M.J.E. Sternberg, Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming, Proc. Nat. Acad. Sci. 93 (1996) 438–442.

[16] R.D. King, S.H. Muggleton, M.J.E. Sternberg, Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase, Proc. Nat. Acad. Sci. 89 (23) (1992) 11322–11326.

[17] S. Kramer, Structural regression-trees, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), MIT Press, 1996, pp. 812–819.

[18] N. Lavrac, S. Dzeroski, ILP: Techniques and Applications, Ellis Horwood, London, 1994.

[19] J.W. Lloyd, Foundations of Logic Programming, Springer, Berlin, 1984.
[20] D. Michie, R.A. Chambers, BOXES: An experiment in adaptive control, in: E. Dale, D. Michie (Eds.), Machine Intelligence 2, Oliver and Boyd, Edinburgh, 1968, pp. 137–152.
[21] F. Mizoguchi, H. Ohwada, An inductive logic programming approach to constraint acquisition for constraint-based problem solving, in: L. De Raedt (Ed.), Fifth International Inductive Logic Programming Workshop, Katholieke Universiteit Leuven, Heverlee, Belgium, 1995.
[22] S. Muggleton, Inductive logic programming, New Gen. Comput. 8 (4) (1991) 295–318.
[23] S. Muggleton, Inductive logic programming: derivations, successes and shortcomings, SIGART Bulletin 5 (1) (1994) 5–11.
[24] S. Muggleton, Inverse entailment and Progol, New Gen. Comput 13 (1995) 245–286.
[25] S. Muggleton, Learning from positive data, in: Proceedings of the Sixth Inductive Logic Programming Workshop, LNAI, Springer, Berlin, 1996, pp. 358–376.
[26] S. Muggleton, R. King, M. Sternberg, Predicting protein secondary structure using inductive logic programming, Protein Engineering 5 (1992) 647–657.
[27] S. Muggleton, C.D. Page, A. Srinivasan, An initial experiment into stereochemistry-based drug design using ILP, in: Proceedings of the Sixth Inductive Logic Programming Workshop, LNAI, Springer, Berlin, 1996.
[28] S. Muggleton, D. Page, Beyond first-order learning: inductive learning with higher-order logic, PRG-TR 13–94, Oxford University Computing Laboratory, Oxford, 1994.
[29] S.H. Muggleton, C. Feng, Efficient induction of logic programs, in: Proceedings of the First Conference on Algorithmic Learning Theory, Ohmsha, Tokyo, 1990.
[30] M.J. Norusis, SPSS: Base System User Guide, Release 6.0, SPSS, 444 N Michigan Ave, Chicago, Illinois 60611, 1994.
[31] University of Stuttgart, Snns user manual, version 4.1, Technical Report 6/95, Institute for Parallel and Distributed High Performance Systems, Stuttgart, 1995.
[32] G.D. Plotkin, Automatic Methods of Inductive Inference, PhD. thesis, Edinburgh University, August 1971.
[33] J.R. Quinlan, Learning logical definitions from relations, Machine Learning 5 (1990) 239–266.
[34] J.R. Quinlan, Learning with continuous classes, in: A. Adams, L. Sterling (Eds.), Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence, Singapore, World Scientific, 1992, pp. 343–348.
[35] J.R. Quinlan, R.M. Cameron-Jones, Efficient top-down induction of logic programs, SIGART Bulletin 5 (1) (1994) 33–42.
[36] L. De Raedt, L. Dehaspe, Clausal discovery, Technical Report CW 238, Katholieke Universiteit Leuven, Heverlee, Belgium, 1996.
[37] L. De Raedt, L. Dehaspe, Clausal discovery (special issue on ILP), Machine Learning 26 (1997) 99–146.
[38] L. DeRaedt, S. Dzeroski, First order jk-clausal theories are PAC-learnable, Artificial Intelligence 70 (1994) 375–392.
[39] C.A. Ramsden, Comprehensive Medicinal Chemistry, vol. 4, Pergamon Press, Oxford, 1979.
[40] J.O. Rawlings, Applied Regression Analysis: A Research Tool, Wadsworth and Brooks, Pacific Grove, California, 1988.
[41] M. Sebag, C. Rouveirol, Constraint inductive Logic Programming, in: L. De Raedt (Ed.), Fifth International Inductive Logic Programming Workshop, Katholieke Universiteit Leuven, Heverlee, Belgium, 1995.
[42] E.Y. Shapiro, Algorithmic Program Debugging, MIT Press, 1983.
[43] A. Srinivasan, R.C. Camacho, Experiments in numerical reasoning with ILP, Technical Report PRG-TR-22-96, Oxford University Computing Laboratory, Oxford, 1996.
[44] A. Srinivasan, R.D. King, Feature construction with inductive logic programming: a study of quantitative predictions of biological activity aided by structural attributes, in: Proceedings of the Sixth Inductive Logic Programming Workshop, LNAI, Springer, Berlin, 1996.
[45] A. Srinivasan, S.H. Muggleton, R.D. King, M.J.E. Sternberg, Mutagenesis: ILP experiments in a non-determinate biological domain, in: S. Wrobel (Ed.), Proceedings of the Fourth International Inductive Logic Programming Workshop, Gesellschaft fur Mathematik und Datenverarbeitung MBH, GMD-Studien Nr 237, 1994.
[46] R.E. Walpole, R.H. Myers, Probability and Statistics for Engineers and Scientists, 2nd ed., Collier Macmillan, New York, 1978.

[47] S.M. Weiss, C.A. Kulikowski, Computer Systems that Learn, Morgan Kaufmann, San Mateo, CA, 1991.
[48] D.M. Young, R.T. Gregory, A Survey of Numerical Mathematics, vol. 1, Addison–Wesley, Reading, MA, 1972.
[49] J. Zelle, R. Mooney, Learning semantic grammars with constructive inductive logic programming, in: Proceedings of the Eleventh National Conference on Artificial Intelligence, Morgan Kaufmann, 1993, pp. 817–822.